Моделирование и анализ информационных систем. Т. 24, № 4 (2017), с. 415–433
Modeling and Analysis of Information Systems. Vol. 24, No 4 (2017), pp. 415–433

DOI: 10.18255/1818-1015-2017-4-415-433

УДК 517.9

О задаче минимизации последовательных программ

Захаров В. А.[1], **Жайлауова Ш. Р.**

получена 17 июля 2017

Аннотация. Стандартные схемы программ — это одна из наиболее простых моделей последовательных императивных программ, предназначенная для решения задач оптимизации и верификации программ. Мы рассматриваем разрешимое отношение логико-термальной эквивалентности стандартных схем программ и задачу минимизации их размера при условии сохранения отношения логико-термальной эквивалентности. Нами доказано, что эта задача является алгоритмически разрешимой. Далее показано, что стандартные схемы программ с отношением логико-термальной эквивалентности и конечные детерминированные автоматы-преобразователи, работающие над полугруппами подстановок, взаимно транслируются друг в друга. Отсюда следует, что также разрешимы задачи проверки эквивалентности и минимизации для преобразователей указанного вида. Кроме того, на основе обнаруженной взаимосвязи выделен подкласс стандартных схем программ, минимизация которых осуществима за полиномиальное время при помощи ранее известных методов минимизации автоматов-преобразователей, работающих над полугруппами. В заключении приведен пример, свидетельствующий о том, что в общем случае задача минимизации автоматов-преобразователей над полугруппой подстановок может иметь несколько неизоморфных решений.

Ключевые слова: последовательная программа, автомат-преобразователь, минимизация, подстановка, полугруппа, эквивалентность

Для цитирования: Захаров В. А., Жайлауова Ш. Р., "О задаче минимизации последовательных программ", *Моделирование и анализ информационных систем*, **24**:4 (2017), 415–433.

Об авторах:
Захаров Владимир Анатольевич, orcid.org/0000-0002-3794-9565,
д-р физ.-мат. наук, ст. науч. сотр.,
Национальный исследовательский университет "Высшая школа экономики", факультет компьютерных наук,
ул. Мясницкая, 20, г. Москва, 101000 Россия, e-mail: zakh@cs.msu.ru

Жайлауова Шынар Рустамбековна, магистр,
Московский государственный университет им. М.В. Ломоносова, факультет ВМК,
Ленинские горы, д. 1, стр. 52, ГСП-1, Москва, 119991 Россия, e-mail: gulgaisha93@mail.ru

Благодарности:
[1]Работа выполнена при финансовой поддержке грантов РФФИ №16-01-00546 и №15-01-05742.

Введение

Теория схем программ — первая математическая теория, объектами изучения которой являются компьютерные программы (не путать с алгоритмами!), — возникла в конце 50-х годов с целью создания математических методов, облегчающих

разработку программ. Компьютеры того времени обладали очень ограниченными вычислительными ресурсами, и поэтому задача оптимизации программ по размеру, объему рабочей памяти и производительности была чрезвычайно актуальной. Часто отлаженную и протестированную программу нужно было модифицировать вручную, чтобы сократить ее размер или оптимизировать рабочую память. После проведения оптимизирующих преобразований измененная программа должна была вновь подвергаться тестовым испытаниям. Чтобы облегчить этот процесс, А.А. Ляпунов в статье [12] предложил следующий подход к решению задачи оптимизации программ:

1) создать математическую модель, в которой компьютерные программы могли быть представлены выражениями, подобными логическим или алгебраическим формулам;

2) определить поведение (семантику) моделей программ так, чтобы эквивалентные модели программ (т.е. модели, имеющие одинаковое поведение) соответствовали программам, вычисляющим одну и ту же функцию;

3) использовать для оптимизации программ только такие преобразования, которые сохраняют эквивалентность предложенных моделей программ.

Модели программ, предложенные в статье [12], были названы схемами программ по аналогии с терминами «схема формулы» или «схема правила вывода», используемыми в математической логике. Позднее Р.И. Подловченко предложила назвать модели программ, удовлетворяющие требованию п. 2, аппроксимирующими моделями [13].

Если использовать указанный подход, то для подтверждения корректности программной оптимизации можно не проводить повторного тестирования оптимизированной программы, а всего лишь проверить эквивалентность схемы исходной программы и схемы модифицированной программы. При наличии алгоритма проверки эквивалентности схем программ эту проверку можно автоматизировать, и поэтому на первый план была выдвинута задача создания алгоритмов проверки эквивалентности схем программ. Она была успешно решена Ю.И. Яновым в работе [14]. Достигнутый успех привлек к теории схем программ внимание со стороны математиков, занимавшихся решением задач системного программирования. Спустя короткое время на основе этой теории были предложены алгоритмы оптимизации линейных участков программ [10], рабочей памяти программ [2, 11]; эти и другие подобные алгоритмы были использованы в первом оптимизирующем трансляторе для языка программирования ALGOL-60 [16]. Первые положительные результаты исследования алгоритмических задач в теории схем программ показали, что с помощью этой теории можно создавать эффективные методы оптимизации программ.

Другое важное открытие в теории схем программ совершила Дж. Ратледж [22]; она показала, что схемы программ Ляпунова–Янова и конечные автоматы Рабина–Скотта можно взаимно транслировать друг в друга — вычисления схем программ можно представить как вычисления автоматов и наоборот. Отсюда следует, что разрешающие алгоритмы для задач теории автоматов можно перенести в теорию схем программ. В частности, многочисленные алгоритмы минимизации детерминированных конечных автоматов [26] можно использовать для сокращения размера

программ. На основании обнаруженной взаимосвязи теории схем программ и теории автоматов был предложен следующий метод построения алгоритмов оптимизации программ (см. [3]). Предположим, что имеется аппроксимирующая модель программ $\mathcal{P}$ с отношением эквивалентности $\sim_{\mathcal{P}}$ и мерой сложности $||\cdot||_{\mathcal{P}}$ на множестве схем программ. Предположим также, что удалось обнаружить класс автоматов $\mathcal{A}$ с отношением эквивалентности $\sim_{\mathcal{A}}$ и мерой сложности $||\cdot||_{\mathcal{A}}$, для которого существует эффективная взаимнооднозначная трансляция $\varphi : \mathcal{P} \to \mathcal{A}$, удовлетворяющая двум условиям:

1) для любой пары схем программ π_1, π_2 справедливо соотношение $\pi_1 \sim_{\mathcal{P}} \pi_2 \Leftrightarrow \varphi(\pi_1) \sim_{\mathcal{A}} \varphi(\pi_2)$;

2) для любой схемы программ π верно равенство $||\pi||_{\mathcal{P}} = ||\varphi(\pi)||_{\mathcal{A}}$.

При соблюдении этих условий любой алгоритм минимизации автоматов из класса $\mathcal{A}$ в метрике сложности $||\cdot||_{\mathcal{A}}$ можно легко адаптировать для решения задачи минимизации программ в метрике сложности $||\cdot||_{\mathcal{P}}$. Здесь теория автоматов предоставляет абстрактную модель вычислений, в которой удобно решать проблемы эквивалентности и минимизации, а теория схем программ — средство для приложения полученных результатов в решении задач анализа и оптимизации программ.

Помимо работы [22], описанный прием был успешно применен А.А. Летичевским в работе [1] для решения задачи оптимизации программ по быстродействию; в качестве схем программ использовались дискретные преобразователи Глушкова, обобщающие схемы программ Ляпунова–Янова. Достигнутый успех стимулировал поиск других взаимосвязей между теорией схем программ и теорией автоматов. В статье [19] было показано, что схемы последовательных программ с перестановочными операторами можно транслировать в детерминированные многоленточные автоматы, а в статье [17] было установлено, что подобная же трансляция имеет место для рекурсивных схем программ и детерминированных автоматов с магазинной памятью. Однако проблема эквивалентности для этих двух разновидностей автоматов оказалась твердым орешком — лишь два десятилетия спустя алгоритмическая разрешимость этой проблемы была доказана в статьях [18, 24] (вопрос о ее сложности открыт до сих пор). Большой интерес специалистов в теории схем программ, проявленный к проблеме эквивалентности для многоленточных и магазинных автоматов, оказал отрицательное влияние на развитие этой теории — были обойдены вниманием другие виды схем программ и классы автоматов, для которых осуществима описанная выше трансляция, не удалось построить практичных алгоритмов проверки эквивалентности и оптимизации схем программ. В итоге намеченные в статье [3] цели развития теории схем программ и ее применения в системном программировании не были своевременно достигнуты. Эти неудачи послужили одной из причин последующего упадка этой теории; более подробно об этом периоде развития теории схем программ можно прочесть в книге [7].

И хотя в настоящее время задачи, исследовавшиеся в теории схем программ в 70–80 гг., уже не относятся к числу наиболее актуальных, потенциал некоторых моделей и методов, созданных и развитых в рамках этой теории, еще далек от исчерпания. В частности, мы предлагаем исследовать еще одну сочетаемую пару «схема программ — автомат», пригодную для решения задачи оптимизации последовательных императивных программ. В качестве аппроксимирующей модели программ $\mathcal{P}$

рассматривается модель стандартных схем программ, впервые предложенная в статье [19], с отношением логико-термальной эквивалентности $\sim_{\mathcal{P}}$, впервые описанным в статье [8]. Выбор этой модели программ обусловлен тем, что

1) проблема логико-термальной эквивалентности стандартных схем программ разрешима за полиномиальное время (см. [4, 23]);

2) семантику стандартных схем программ можно описать при помощи конечных подстановок, образующих полугруппу, которая обладает многими полезными алгебраическими свойствами и хорошо подходит для решения задачи минимизации автоматов.

Как будет показано далее в данной статье, выбранной модели программ соответствует класс конечных автоматов-преобразователей над полугруппами, введенный в статье [28]. Для автоматов этого вида эффективно разрешимы проблема проверки эквивалентности [28] и проблема минимизации [6].

В данной статье представлены три основных результата исследования проблемы минимизации стандартных схем программ. Во-первых, показано, что задача минимизации стандартных схем программ с логико-термальной эквивалентностью алгоритмически разрешима и сводится к задаче минимизации конечных автоматов-преобразователей над полугруппой конечных подстановок. Во-вторых, выделена подполугруппа консервативных ортогональных подстановок, для которой эффективно разрешима задача минимизации автоматов-преобразователей при помощи метода минимизации, описанного в статье [6]. В-третьих, установлено, что в общем случае задача минимизации конечных автоматов-преобразователей над полугруппой произвольных конечных подстановок не имеет однозначного решения подобно тому, как не имеет однозначного решения задача минимизации дизъюнктивных нормальных форм в булевой алгебре.

Статья организована следующим образом. Поскольку описание модели стандартных схем программ опирается на понятие подстановки, в следующем разделе приводятся основные понятия алгебры подстановок. Далее следует описание устройства стандартных схем программ и логико-термальной эквивалентности. В следующем разделе доказывается алгоритмическая разрешимость задачи минимизации стандартных схем программ с отношением логико-термальной эквивалентности. Затем дается описание модели вычислений конечных автоматов-преобразователей над полугруппами и показывается, что классы стандартных схем программ конечных детерминированных автоматов-преобразователей над полугруппой подстановок взаимно транслируются друг в друга. Отсюда следует, что задачи проверки эквивалентности и минимизации для преобразователей указанного вида также разрешимы. В следующем разделе выделена подполугруппа консервативных ортогональных подстановок и показано, что задача минимизации автоматов-преобразователей, работающих над этой подполугруппой, имеет однозначное решение, вычислимое за полиномиальное время. Отсюда следует, что для класса схем программ, соответствующих преобразователям указанного вида, существует эффективное решение задачи минимизации. В заключении приводится пример автомата-преобразователя над полугруппой подстановок, для которого задача минимизации имеет несколько неизоморфных решений. Этот пример призван обозначить некоторые трудности,

с которыми придется столкнуться при изучении сложности задачи минимизации стандартных схем программ.

1. Алгебра подстановок

Для произвольных заданных множеств переменных X, функциональных символов $\mathcal{F}$ и предикатных символов $\mathcal{P}$ будем использовать запись $Term(X, \mathcal{F})$ для обозначения множества термов, построенных из переменных и функциональных символов, а запись $Atom(X, \mathcal{F}, \mathcal{P})$ — для обозначения множества атомарных формул (атомов), построенных из предикатных символов и термов. Для каждого терма или атома T обозначим записью Var_T множество переменных, входящих в выражение T. Высота терма или атома определяется обычным образом как высота дерева в древесном представлении этих выражений.

Пусть X и Y — два конечных множества переменных. Подстановкой назовем всякое отображение $\theta : X \to Term(Y, \mathcal{F})$, сопоставляющее каждой переменной из X некоторый терм из $Term(Y, \mathcal{F})$. Множество всех таких подстановок условимся обозначать записью $Subst(X, Y, \mathcal{F})$. Если $X = \{x_1, \ldots, x_n\}$ — конечное множество переменных и $\theta(x_i) = t_i$ для всех i, $1 \le i \le n$, то такая подстановка называется конечной и определяется списком пар $\{x_1/t_1, \ldots, x_n/t_n\}$. Запись Var_θ будет использоваться для обозначения множества переменных $\bigcup\limits_{i=1}^{n} Var_{t_i}$, входящих в состав всех термов t_i подстановки θ. Результатом применения подстановки θ к терму t является терм $t\theta$, получающийся одновременной заменой в терме t каждого вхождения любой переменной x_i термом t_i. Композиция $\eta \circ \theta$ подстановок $\theta \in Subst(X, Y, \mathcal{F})$ и $\eta \in Subst(Y, Z, \mathcal{F})$ — это такая подстановка из множества $Subst(X, Z, \mathcal{F})$ (ее традиционно обозначают записью $\theta\eta$), которая определяется равенством $\theta\eta(x) = \theta(x)\eta$ для каждой переменной x, $x \in X$. Множество подстановок $Subst(X, X, \mathcal{F})$ с операцией композиции образует полугруппу, в которой нейтральным элементом служит тождественная подстановка $\varepsilon = \{x_1/x_1, \ldots, x_n/x_n\}$.

На множестве подстановок $Subst(X, X, \mathcal{F})$ можно определить отношения предпорядка $\preceq$. Для пары подстановок θ_1, θ_2 отношение $\theta_1 \preceq \theta_2$ выполняется, если есть такая подстановка η, что $\theta_2 = \theta_1\eta$. В случае выполнимости отношения $\theta_1 \preceq \theta_2$ подстановку θ_1 называют прототипом подстановки θ_2, а подстановку θ_2 — примером подстановки θ_1. Квазиупорядоченное множество $(Subst(X, X, \mathcal{F}), \preceq)$ образует квазирешетку, наименьшим элементом которой является тождественная подстановка ε. Эта квазирешетка удовлетворяет условию обрыва убывающих цепей. Более подробные сведения об алгебре подстановок могут быть почерпнуты из работы [15].

2. Стандартные схемы программ

В качестве математической модели последовательных императивных программ мы используем стандартные схемы программ; они были введены в статье [19] и получили такое название в статье [3]. Подробное описание этой разновидности схем программ приведено в монографии [9]. Мы опишем вкратце устройство этой модели, опираясь на понятия теории графов и алгебры подстановок. Пусть задано

некоторое конечное множество переменных X. Стандартная схема программ π представляет собой размеченный ориентированный граф; его вершины будем называть узлами. Каждому узлу v графа π приписана атомарная формула A_v из множества $Atom(X, \mathcal{F}, \mathcal{P})$. Один из узлов v_0 графа π особо выделен в качестве входа в программу, а некоторые другие узлы $u_1, \ldots, u_k$ являются выходами из программы. Из каждого невыходного узла исходят не более двух дуг, помеченных разными символами из множества $\{0, 1\}$. Кроме того, каждой дуге, ведущей в графе π из узла u в узел v, приписана подстановка θ_{uv} из множества $Subst(X, X, \mathcal{F})$. Из выходов схемы не исходит ни одной дуги. Предполагается также, что через каждый узел графа π проходит некоторый маршрут, ведущий из входа схемы в один из ее выходов. Размером $||\pi||$ схемы программ π назовем число ее узлов.

Узлы графа π соответствуют точкам программы, в которых проводится проверка условий и ветвление потока управления программы. Атомарные формулы, приписанные узлам, соответствуют условиям ветвления, а дуги графа π — линейным участкам программы. Если линейный участок между точками ветвления u и v представляет собой последовательность операторов присваивания $x_{i_1} := t_1; \ldots; x_{i_m} := t_m$, то соответствующей дуге приписана подстановка $\theta_{uv} = \{x_{i_m}/t_m\} \cdots \{x_{i_1}/t_1\}$, представляющая собой композицию односвязочных подстановок для всех операторов этого линейного участка.

Пусть задан некоторый маршрут в схеме программ π из узла v_0 в узел v_m

$$w = v_0 \xrightarrow{\sigma_1,\theta_1} v_1 \xrightarrow{\sigma_2,\theta_2} \cdots \xrightarrow{\sigma_m,\theta_m} v_m \ . \tag{1}$$

Если v_0 — вход схемы π, то маршрут w называется *начальным*, а если v_m — один из выходов схемы π, то маршрут w называется *финальным*. Маршрут w, который является одновременно начальным и финальным, называется *полным маршрутом*. Множество всех полных маршрутов в схеме программ π обозначим записью $Path(\pi)$. Записью θ_w будем обозначать композицию $\theta_1 \circ \cdots \circ \theta_m$ всех подстановок, приписанных дугам этого маршрута.

Двоичная последовательность $lh(w) = \sigma_1, \sigma_2, \ldots, \sigma_m$ называется *логической историей* маршрута (1); вместе с узлом v_0 она однозначно определяет маршрут w в схеме программ π. Последовательность пар

$$lth(w) = (A_{v_0}\eta_0, \sigma_1), (A_{v_1}\eta_1, \sigma_2), \ldots, (A_{v_{m-1}}\eta_{m-1}, \sigma_m), (A_{v_m}\eta_m, 1),$$

где $\eta_0 = \varepsilon$ и $\eta_i = \theta_i\eta_{i-1}$ для каждого i, $1 \le i \le m$, называется *логико-термальной историей* (л-т историей) маршрута w. *Детерминантом* $Det(\pi)$ схемы программ π называется множество л-т историй всех полных маршрутов в схеме программ π, т. е. $Det(\pi) = \{lth(w) \ : \ w \in Path(\pi)\}$. Две схемы программы π_1 и π_2 считаются *логико-термально (л-т) эквивалентными* (отношение обозначается записью $\pi_1 \sim_{lt} \pi_2$), если справедливо равенство $Det(\pi_1) = Det(\pi_2)$.

Схема программ π' называется *S-минимальной*, если неравенство $||\pi'|| \le ||\pi||$ выполняется для любой схемы программ π, л-т эквивалентной схеме π'. Задача минимизации схем программ состоит в том, чтобы для произвольной заданной схемы программ π построить л-т эквивалентную ей минимальную схему программ π'.

Функциональная эквивалентность стандартных схем программ определяется на множестве всех интерпретаций сигнатуры $\Sigma = (X, \mathcal{F}, \mathcal{P})$. Для каждой интерпре-

тации I в схеме программ π выбирается единственный (если он существует) полный маршрут $w(\pi, I)$, который в каждом узле v_i, $0 \le i < m$, удовлетворяет условию $I \models A_{v_i}\eta_i = \sigma_{i+1}$. Тогда атомарная формула $A_{v_m}\eta_m$ объявляется значением $\pi(I)$ схемы программ π в интерпретации I. Две схемы программ π_1 и π_2 считаются функционально эквивалентными, если для любой интерпретации I выполняется соотношение $I \models \pi_1(I) = \pi_2(I)$.

В статье [19] было показано, что отношение функциональной эквивалентности алгоритмически неразрешимо, но, как было установлено в работе [8], оно аппроксимируется разрешимым отношением л-т эквивалентности. Таким образом, алгоритмы минимизации стандартных схем программ с л-т эквивалентностью можно корректно использовать для оптимизации последовательных императивных программ. Далее мы покажем, что указанная задача минимизации стандартных схем программ с л-т эквивалентностью алгоритмически разрешима и, кроме того, сводима к задаче минимизации конечных автоматов-преобразователей над полугруппой подстановок. Для этого целесообразно обратиться к следующему простому критерию л-т эквивалентности схем программ.

Сокращенной логико-термальной историей маршрута (1) в схеме программ π назовем пару $rlth(w) = (lh(w), A_{v_m}\theta_w)$, состоящую из логической истории $lh(w) = \sigma_1, \sigma_2, \ldots, \sigma_m$ этого маршрута и примера атомарной формулы A_{v_m}, приписанной последнему узлу v_m маршрута w, которая специализирована композицией подстановок $\theta_1 \circ \cdots \circ \theta_m$, приписанных всем дугам этого маршрута. *Сокращенным детерминантом* схемы программ π назовем множество

$$RDet(\pi) = \{rlth(w) \ : \ w \text{ — начальный маршрут в схеме } \pi\}$$

сокращенных л-т историй всех начальных маршрутов в схеме программ π.

Утверждение 1. *Для любой пары схем программ π_1 и π_2, через каждый узел которых проходит хотя бы один полный маршрут, справедливо соотношение*

$$\pi_1 \sim_{lt} \pi_2 \ \Leftrightarrow \ RDet(\pi_1) = RDet(\pi_2) \ .$$

Доказательство. Достаточно заметить, что в случае, когда через каждый узел схемы программ проходит хотя бы один полный маршрут, каждой л-т истории $lth(w) = (A_{v_0}\eta_0, \sigma_1), (A_{v_1}\eta_1, \sigma_2), \ldots, (A_{v_{m-1}}\eta_{m-1}, \sigma_m), (A_{v_m}\eta_m, 1)$ взаимнооднозначно соответствует набор, состоящий из $m + 1$ сокращенных л-т историй $rlth(w_i) = (\sigma_1, \ldots, \sigma_i, A_{v_i}\eta_i), 0 \le i \le m$. □

3. Минимизация стандартных схем программ

Как уже было упомянуто, отношение л-т эквивалентности стандартных схем программ разрешимо (см. [4, 8, 23]). Если в модели вычислений разрешима проблема эквивалентности, то для решения задачи минимизации вычислительных систем (программ) в этой модели можно использовать переборный алгоритм: для минимизации заданной программы π нужно перебирать в порядке возрастания сложности все программы π' и проверять эквивалентность $\pi \sim \pi'$. Если для каждого значения

сложности k существует лишь конечное множество программ π', имеющих сложность $||\pi'|| = k$, то мы получаем готовый (хотя и далеко не самый эффективный) алгоритм минимизации программ в рассматриваемой модели вычислений.

К сожалению, если мерой сложности стандартной схемы программ π является число ее узлов, то для каждого натурального $k, k > 1$, существует бесконечно много схем программ сложности k. Причина тому — неограниченность множества подстановок, которыми можно помечать дуги программы. Чтобы преодолеть это препятствие, можно попытаться для каждой схемы программы π выделить такое конечное подмножество подстановок, которого достаточно для построения минимальной схемы программ, эквивалентной схеме π.

Пусть задана схема программ π с множеством узлов V, дуги которой помечены конечными подстановками из множества $Subst(X, X, \mathcal{F})$, где $|X| = n$. Мы будем полагать, что каждый узел схемы π лежит на некотором маршруте из входа схемы в один из ее выходов. Как можно легко заметить, не удовлетворяющие этому требованию узлы можно удалить вместе с инцидентными им дугами, сохранив при этом детерминант $Det(\pi)$ схемы. Рассмотрим произвольный узел v в этой схеме и переменную x. Переменная x считается *существенной* в узле v схемы π, если из этого узла исходит хотя бы один такой маршрут w, сокращенная л-т история $rlth(w) = (lh(w), A)$ которого удовлетворяет условию $x \in Var_A$. В противном случае переменную x будем называть *несущественной* в узле v.

Как показывает следующее утверждение, для несущественных переменных вид подставляемых термов в соответствующих связках тех подстановок, которыми помечены дуги схемы программы, не имеет значения.

Утверждение 2. *Предположим, что схема программ π содержит дугу $u \xrightarrow{\sigma,\theta} v$, и подстановка θ' отличается от подстановки θ только связкой для переменной x, т.е. $\theta(y) = \theta'(y)$ для любой переменной $y \in X \setminus \{x\}$. Предположим, что схема программ π' получена из π заменой дуги $u \xrightarrow{\sigma,\theta} v$ на дугу $u \xrightarrow{\sigma,\theta'} v$. Тогда если переменная x является несущественной в узле v, то $\pi \sim \pi'$.*

Доказательство. Рассмотрим произвольный маршрут $w = w_1; u \xrightarrow{\sigma,\theta} v; w_2$, проходящий в схеме π через выделенную дугу, и сокращенные л-т истории $rlth(w) = (lh(w), A)$ и $rlth(w_2) = (lh(w_2), A_2)$. Из определения л-т истории маршрутов в схеме программ следует, что $A = A_2\theta\theta_{w_1}$. Поскольку переменная x несущественна в узле v, атом A_2 не содержит вхождений переменной x. А так как подстановки θ и θ' отличаются только связкой для переменной x, верно равенство $A_2\theta = A_2\theta'$. Таким образом, маршрут $w' = w_1; u \xrightarrow{\sigma,\theta'} v; w_2$ будет иметь ту же самую сокращенную л-т историю, что и маршрут w. Значит, $RDet(\pi) = RDet(\pi')$, и поэтому согласно утверждению 1 схемы π и π' л-т эквивалентны. □

Свойство существенности переменной x в узле v схемы программ π можно описать иным способом при помощи графа зависимости переменных в программе, использующегося при анализе потоков данных (см. [21]). Рассмотрим ориентированный граф D_π, вершинами которого служат все пары (x, v), $x \in X$, $v \in V$. В этом графе из вершины (y, u) в вершину (x, v) исходит дуга в том и только том случае, если в схеме программ π есть такая дуга $u \xrightarrow{\sigma,\theta} v$, что $y \in Var_{\theta(x)}$. В определен-

ном таким образом графе зависимости переменных выделим множество вершин $U_\pi = \{(x, v) : x \in Var_{A_v}\}$.

Утверждение 3. *Переменная x существенна в узле v схемы программ π тогда и только тогда, когда в графе зависимости переменных D_π существует путь из вершины (x, v) в одну из вершин множества U_π.*

Доказательство. Утверждения такого рода хорошо известны в теории статического анализа программ. В данном случае оно легко доказывается индукцией по длине маршрута в схеме программ π, подтверждающего существенность переменной x в узле v (необходимое условие), и по длине пути в графе зависимости D_π из вершины (x, v) в вершину из множества U_π (достаточное условие). □

Используя граф зависимости, покажем, как оценить высоту термов, присутствующих в разметках дуг в эквивалентных схемах программ.

Утверждение 4. *Предположим, что схемы программ π_1 и π_2 над множеством переменных X, где $|X| = n$, обладают следующими свойствами:*

1. *$||\pi_1|| = m$, и глубина всех термов в подстановках, которыми помечены дуги в схеме π_1, не превосходит некоторого натурального числа k;*
2. *$||\pi_2|| \le m$, и в схеме π_2 есть такая дуга и $\xrightarrow{\sigma,\eta} v$, что переменная x является существенной в узле v, а высота терма $\eta(x)$ превосходит величину $km(n+1)$.*

Тогда $\pi_1 \not\sim_{lt} \pi_2$.

Доказательство. Согласно утверждению 3 существенность переменной x в узле v влечет за собой существование пути в графе зависимости переменных D_{π_2} из вершины (x, v) в одну из вершин множества U_{π_2}. Поскольку $||\pi_2|| \le m$ и $|X| = n$, кратчайший путь такого вида имеет длину, не превосходящую величины mn. Тогда в схеме программ π_2 существует соответствующий этому пути маршрут w'' такой же длины из узла v в некоторый узел v'', который подтверждает существенность переменной x в узле v. Последнее означает, что $rlth(w'') = (lh(w''), A)$ и $x \in Var_A$. Рассмотрим в схеме программ π_2 кратчайший маршрут w' из входа схемы в вершину u. Длина этого маршрута не превосходит числа m. Тогда начальный маршрут $w_2 = w', u \xrightarrow{\sigma,\eta} v, w''$ имеет длину, меньшую величины $m(n+1)$, и сокращенную л-т историю $rlth(w_2) = (lh(w_2), A\eta\theta_{w'})$. Поскольку $x \in Var_A$ и высота терма $\eta(x)$ превосходит величину $km(n+1)$, высота атома $A\eta\theta_{w'}$ также превосходит эту величину.

Обратимся теперь к схеме программ π_1 и рассмотрим в ней произвольный начальный маршрут w_1, длина которого не превосходит величины $m(n+1)$. Сокращенная л-т история этого маршрута имеет вид $rlth(w_1) = (lt(w_1), B\theta_{w_1})$, причем подстановка θ_{w_1} представляет собой композицию не более чем $m(n+1)$ подстановок, в которых все подставляемые термы имеют высоту, не превосходящую числа k. Значит, высота атома $B\theta_{w_1}$ не превосходит величины $km(n+1)$. Отсюда следует, что $rlth(w_2) \notin RDet(\pi_1)$. Согласно утверждению 1 это означает, что $\pi_1 \not\sim_{lt} \pi_2$. □

На основе приведенных выше утверждений может быть доказана

Теорема 1. *Проблема минимизации стандартных схем программ алгоритмически разрешима.*

Доказательство. Предположим, что требуется минимизировать схему программ π размера $||\pi|| = m$ над множеством переменных X, где $|X| = n$, причем все термы в подстановках, которыми помечены дуги этой схемы, имеют глубину, не превосходящую некоторого натурального числа k. Тогда в минимальной схеме программ π_0, л-т эквивалентной схеме программ π, для каждой дуги $u \xrightarrow{\sigma,\eta} v$ и для каждой переменной x верно одно из двух:

- если переменная x несущественна в узле v, то согласно утверждению 2 терм $\eta(x)$ можно задавать произвольно, сохраняя при этом л-т эквивалентность схемы программ; поэтому можно полагать, что в этом случае $\eta(x) = x$;
- если переменная x существенна в узле v, то согласно утверждению 3 высота терма $\eta(x)$ не превосходит величины $km(n+1)$.

Поскольку проблема л-т эквивалентности стандартных схем программ разрешима, поиск минимальной схемы программ π_0 можно проводить полным перебором по конечной совокупности всех схем программ, размер которых меньше числа m, и при этом таких, что во всех подстановках, приписанных дугам схемы, все термы имеют высоту, не превосходящую величины $km(n+1)$. □

Конечно, такой переборный алгоритм минимизации совершенно непрактичный — он имеет, по меньшей мере, тройную экспоненциальную сложность. Далее мы обсудим перспективы создания более эффективных способов минимизации стандартных схем программ при помощи методов минимизации других систем вычислений.

4. Автоматы-преобразователи над полугруппами

Пусть заданы два конечных множества $\mathcal{C}$ и $\mathcal{A}$. Элементы множества $\mathcal{C}$ будем называть *входными сигналами*. Конечные последовательности входных сигналов (слова в алфавите $\mathcal{C}$) будем называть *потоками сигналов*. Обозначим записью $\mathcal{C}^*$ множество всевозможных потоков сигналов. Элементы множества $\mathcal{A}$ будем называть *простыми действиями*. Конечные последовательности простых действий, представляющие собой слова в алфавите $\mathcal{A}$, будем называть *составными действиями*. Рассмотрим полугруппу $(S, e, \circ)$, порожденную множеством простых действий $\mathcal{A}$, в которой e обозначает нейтральный элемент, а $\circ$ — полугрупповую операцию. Элементы полугруппы S назовем *состояниями данных*. Применив простое действие a к состоянию данных s, получаем результат $s \circ a$. Составное действие $h = a_1 a_2 \dots a_k$ представляет собой композицию $[h]_S = a_1 \circ a_2 \circ \dots \circ a_k$ простых действий системы. Индекс S может быть опущен, если из контекста понятно, о какой полугруппе идет речь.

Детерминированный *автомат-преобразователь* с конечным числом состояний над множеством сигналов $\mathcal{C}$ и множеством базовых действий $\mathcal{A}$ — это размеченная система переходов $\Pi = (\mathcal{C}, \mathcal{A}, Q, q_0, F, T, h_0, E)$, состоящая из

- конечного множества *состояний управления* Q,
- *начального состояния* $q_0 \in Q$,
- множества *состояний выхода* $F \subseteq Q$,
- *функции переходов* $T : Q \times \mathcal{C} \to Q \times \mathcal{A}^*$,
- *инициализирующего действия* $h_0 \in \mathcal{A}^*$,
- *функции финализации* $E : F \to \mathcal{A}^*$.

Каждая четверка (q, c, q', h), удовлетворяющая равенству $T(q, c) = (q', h)$, называется *переходом* и обозначается записью $q \xrightarrow{c,h} q'$. *Размером* $||\Pi||$ автомата-преобразователя Π называется число $|Q|$ его состояний управления.

Прогоном автомата-преобразователя Π на потоке сигналов $\alpha = c_1 c_2 \dots c_n$ из состояния управления q называется последовательность переходов

$$q \xrightarrow{c_1,h_1} q_1 \xrightarrow{c_2,h_2} q_2 \xrightarrow{c_3,h_3} \cdots \xrightarrow{c_m,h_m} q' \ , \tag{2}$$

которую будем обозначать записью $q \xrightarrow{\alpha,h}_* q'$, где $h = h_1 h_2 \dots h_m$. Если состояние управления q является инициальным, то прогон также называется *инициальным*, а если $q' \in F$, то прогон называется *финальным*. Прогон, являющийся одновременно инициальным и финальным, называется *полным*. Для полного прогона $q \xrightarrow{\alpha,h}_* q'$ автомата-преобразователя Π, действия которого интерпретируются в полугруппе S, состояние данных $[h_0 h E(q')]_S$ считается *результатом* этого прогона. Поведение автомата-преобразователя Π, характеризуется частичной функцией $\Pi : \mathcal{C}^* \to S$; ее значения для потока сигналов α определяются соотношением

$$\Pi(\alpha) = \begin{cases} [h_0 h E(q')], \text{ если преобразователь } \Pi \text{ имеет полный прогон } q \xrightarrow{w,h}_* q', \\ \text{не определено, в противном случае.} \end{cases}$$

Для заданной полугруппы $(S, e, \circ)$ преобразователи Π_1 и Π_2 называются *S-эквивалентными*, если равенство $\Pi_1(\alpha) = \Pi_2(\alpha)$ выполняется для любого потока сигналов α. Отношение S-эквивалентности условимся обозначать записью $\Pi_1 \sim_S \Pi_2$. Автомат-преобразователь Π' называется *S-минимальным*, если неравенство $||\Pi'|| \leq ||\Pi||$ выполняется для любого преобразователя Π, S-эквивалентного преобразователю Π'. Задача минимизации автоматов-преобразователей над полугруппой S состоит в том, чтобы для произвольного заданного преобразователя Π построить S-эквивалентный ему S-минимальный преобразователь Π'.

Взаимосвязь между схемами программ и автоматами-преобразователями обеспечивается следующими двумя утверждениями.

Утверждение 5. *Существует такая эффективная трансляция φ стандартных схем программ в автоматы-преобразователи, работающие над полугруппой подстановок, которая удовлетворяет условиям:*

1) *для любой пары схем программ π' и π'' верно соотношение $\pi' \sim_{lt} \pi'' \ \Leftrightarrow \varphi(\pi') \sim_{Subst} \varphi(\pi'')$;*

2) *для любой схемы программ π верно равенство $||\pi|| = ||\varphi(\pi)||$.*

Доказательство. Отображение φ определяется следующим образом. Предположим, что схема программ π сигнатуры $\Sigma = (X, \mathcal{F}, \mathcal{P})$ содержит множество узлов V, и в том числе входной узел v_0. Не ограничивая общности, будем полагать, что каждый узел схемы лежит на некотором маршруте из входа схемы в один из ее выходов. Введем в рассмотрение n новых 1-местных функциональных символов $f_1, \dots, f_n$, где $n = |X|$.

Автомат-преобразователь $\Pi = \varphi(\pi)$ работает над множеством входных сигналов $\mathcal{C} = \{0, 1\}$ и множеством состояний данных $Subst(X, X, \mathcal{F} \cup \{f_1, \dots, f_n\} \cup \mathcal{A})$, имеет множество состояний управления V, совпадающее с множеством состояний выхода,

и начальное состояние v_0. Каждая дуга $u \xrightarrow{\sigma,\theta} v$ в схеме программ π становится переходом в преобразователе $\varphi(\Pi)$. Инициализирующим действием служит тождественная подстановка ε. Функция финализации приписывает каждому состоянию управления v подстановку $\eta_v = \{x_1/f_1(A_v), \ldots, x_n/f_n(A_v)\}$.

Из описания преобразователя Π видно, что между множеством начальных маршрутов в схеме π и множеством полных прогонов в автомате-преобразователе Π существует взаимно-однозначное соответствие: каждому начальному маршруту w с сокращенной л-т историей $rlth(w) = (\alpha, A)$ соответствует полный прогон на потоке сигналов α с результатом $\Pi(\alpha) = \{x_1/f_1(A), \ldots, x_n/f_n(A)\}$. Поэтому из утверждения 1 следует справедливость доказываемого утверждения. □

Утверждение 6. *Существует такая эффективная трансляция ψ автоматов-преобразователей, работающих над множеством входных сигналов $\mathcal{C} = \{0, 1\}$ и полугруппой подстановок, в стандартные схемы программ, которая удовлетворяет следующим условиям:*

1) *для любой пары автоматов-преобразователей Π' и Π'' верно соотношение $\Pi' \sim_{Subst} \Pi'' \Leftrightarrow \psi(\Pi') \sim_{lt} \psi(\Pi'')$;*

2) *для любого автомата-преобразователя Π верно равенство $|\Pi| = |\psi(\Pi)|$.*

Доказательство. Пусть задана полугруппа подстановок $Subst(X, X, \mathcal{F})$ и пусть R — некоторый n-местный предикатный символ, где $n = |X|$, а P — некоторый 0-местный предикатный символ. Рассмотрим произвольный детерминированный автомат-преобразователь $\Pi = (\mathcal{C}, Subst(X, X, \mathcal{F}), Q, q_0, F, T, h_0, E)$, причем, не ограничивая общности, будем полагать, что $h_0 = \varepsilon$, а из состояний выхода не исходит ни одного перехода. Стандартная схема программ $\pi = \psi(\Pi)$ устроена так. Ее узлами являются все состояния управления преобразователя Π, входом — начальное состояние q_0, а выходами — все состояния выхода преобразователя Π. Каждый переход $q' \xrightarrow{\sigma,\theta} q''$ в преобразователе Π становится дугой в схеме программ π. Каждому выходу q, $q \in F$, приписана атомарная формула $A_q = R(x_1, \ldots, x_n)E(q)$, а всем остальным узлам схемы приписан предикат P.

Из описания схемы программ π видно, что между множеством полных прогонов в автомате-преобразователе Π и множеством полных маршрутов в схеме π существует взаимно-однозначное соответствие: каждому полному прогону на потоке сигналов $\alpha = \sigma_1 \ldots \sigma_m$, завершающемуся с результатом η, соответствует полный маршрут в схеме π с л-т историей $lth(w) = (P, \sigma_1), \ldots, (P, \sigma_m), (R(x_1, \ldots, x_n)\eta, 1)$. Отсюда следует справедливость доказываемого утверждения. □

Утверждения 5 и 6 обеспечивают переносимость методов и результатов, полученных в одной из теорий — теории схем программ или теории автоматов-преобразователей над полугруппами, — в другую теорию. Например, из утверждения 6 и ранее полученных результатов о разрешимости за полиномиальное время задачи проверки л-т эквивалентности схем программ (см. [4, 23]) следует

Теорема 2. *Проблема эквивалентности конечных детерминированных автоматов-преобразователей, работающих над полугруппой подстановок, разрешима за полиномиальное время.*

Что касается задачи минимизации, то вопрос о ее сложности для схем программ остается открытым; поэтому на основании теоремы 1 и утверждения 5 справедливо более слабое утверждение

Теорема 3. *Проблема минимизации конечных детерминированных автоматов-преобразователей, работающих над полугруппой подстановок, разрешима.*

В следующем разделе мы покажем, что утверждение 5 о транслируемости автоматов-преобразователей в схемы программ вкупе с эффективными методами минимизации автоматов-преобразователей над полугруппами дает эффективное решение задачи минимизации для некоторых классов схем программ.

5. Минимизация автоматов-преобразователей над полугруппой подстановок

Первый алгоритм минимизации конечных детерминированных автоматов-преобразователей над свободной полугруппой слов был описан в статье [20]. В работе [5] была исследована задача минимизации автоматов-преобразователей, работающих над группами; было показано, что если множество состояний данных S образует группу с разрешимой проблемой тождеств, то задача минимизации автоматов-преобразователей имеет однозначное решение, которое может быть эффективно вычислено. Затем в статье [6] было получено решение задачи минимизации автоматов-преобразователей над упорядоченными полугруппами. Поскольку полугруппа подстановок $Subst(X, X, F)$ является квазирешеткой, метод, предложенный в этой статье, применим для решения задачи минимизации автоматов-преобразователей, работающих над $Subst(X, X, F)$. Рассмотрим требования, предъявляемые в статье [6] к полугруппе S, и оценим, в какой мере полугруппа подстановок $Subst(X, X, F)$ удовлетворяет им.

На полугруппе S определим бинарное отношение $\preceq_S$ следующим образом: для любой пары элементов s_1, s_2 отношение $s_1 \preceq_S s_2$ выполняется тогда и только тогда, когда для некоторого элемента s имеет место равенство $s_1 \circ s = s_2$. Полугруппа называется *упорядоченной*, если $\preceq_S$ является отношением частичного порядка на множестве S. Первое требование, предъявляемое к рассматриваемой полугруппе, таково.

Req1: Частично упорядоченное множество $(S, \preceq)$ является фундированной решеткой, в которой для каждой пары элементов $[h_1]$ и $[h_2]$ эффективно вычислима их точная нижняя грань.

Второе требование касается разрешимости линейных уравнений в рассматриваемой полугруппе $(S, e, \circ)$.

Req2: Существует алгоритм решения уравнений вида $[g] \circ X = [h]$ для любой заданной пары действий $g, h \in \mathcal{A}^*$.

Последнее требование свойства сократимости. Полугруппа называется *левосократимой*, если всякое уравнение вида $s \circ X = \widehat{s}$ имеет не более одного решения, т.е. соотношение $s \circ s' = s \circ s'' \;\Rightarrow\; s' = s''$ верно для любой тройки s, s', s''.

Req3: $(S, e, \circ)$ — левосократимая полугруппа.

Полугруппа подстановок $Subst(X, X, F)$ удовлетворяет требованию **Req1** — она является фундированной квазирешеткой с отношением предпорядка $\preceq$, в которой точная верхняя грань подстановок вычислима при помощи процедуры унификации, а точная нижняя грань вычислима при помощи процедуры антиунификации (устройство этой квазирешетки подробно описано в [15]). Как известно, обе эти процедуры можно выполнить за полиномиальное относительно размера обрабатываемых подстановок время. Эквивалентные подстановки в квазиупорядоченном множестве $(Subst(X, X, F), \preceq)$ отличаются лишь наименованием переменных в подставляемых термах; это обстоятельство не оказывает существенного влияния на алгоритм минимизации и обоснование его корректности, приведенные в статье [6].

Очевидный способ решения уравнений вида $\theta \circ X = \eta$ в алгебре подстановок — это простой перебор всех подстановок, в которых высота подставляемых термов не превосходит максимальной высоты термов в подстановке η. Однако можно заметить, что эта задача полиномиально сводится к проблеме проверки вхождения одного терма в другой в качестве подтерма и поэтому может быть решена за полиномиальное время.

Однако последнее требование не выполняется для произвольных подстановок. Рассмотрим подстановки $\theta = \{x_1/x_1, x_2/f(x_1)\}$, $\theta' = \{x_1/f(x_1), x_2/f(x_1)\}$ и $\theta'' = \{x_1/x_2, x_2/x_2\}$. Нетрудно заметить, что имеет место равенство $\theta \circ \theta' = \theta \circ \theta'' = \{x_1/f(x_1), x_2/f(x_1)\}$, однако, подстановки θ' и θ'' не являются эквивалентными в квазирешетке $(Subst(X, X, F), \preceq)$.

Коль скоро вся полугруппа подстановок не удовлетворяет требованию **Req1**, для применения метода минимизации автоматов-преобразователей, предложенного в статье [6], нужно выделить как можно более обширную подполугруппу подстановок, удовлетворяющих требованиям **Req1**–**Req3**. В качестве такой подполугруппы мы предлагаем использовать множество консервативных ортогональных подстановок, введенных в статье [27].

Подстановка $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ называется *консервативной*, если $Var_\theta = X$, и *ортогональной*, если ни один терм t_i, $1 \le i \le n$, не является подтермом никакого другого терма t_j, $j \ne i$. Например, подстановка $\theta_0 = \{x_1/f(x_2), x_2/x_1\}$ является консервативной и ортогональной, подстановка $\theta_1 = \{x_1/f(x_2), x_2/g(x_2)\}$ не является консервативной, а подстановка $\theta_2 = \{x_1/f(x_1), x_2/h(x_2, f(x_1))\}$ не является ортогональной. Класс всех консервативных и ортогональных подстановок обозначим записью $COSubst(X, X, F)$.

Исходя из приведенного определения, нетрудно заметить, что композиция консервативных ортогональных подстановок также является консервативной ортогональной подстановкой и, кроме того, любой пример и любой шаблон всякой консервативной ортогональной подстановки также является консервативной и ортогональной подстановкой. Поэтому класс $COSubst(X, X, F)$ образует подполугруппу подстановок, а квазиупорядоченное множество подстановок $(COSubst(X, X, F), \preceq)$ является фундированной квазирешеткой, удовлетворяющей требованию **Req1**.

Утверждение 7. *Полугруппа консервативных ортогональных подстановок удовлетворяет свойству левого сокращения.*

Доказательство. Если бы для некоторой пары различных термов t', t'' и ортогональной подстановки $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ выполнялось равенство $t'\theta = t''\theta$, то и для некоторой переменной x_i и отличного от нее терма t также выполнялось равенство $x_i\theta = t\theta$. Но это означало бы, терм t_i содержит в качестве подтермов некоторые термы из множества $\{t_1, \ldots, t_n\}$ вопреки определению свойства ортогональности подстановки. □

Таким образом, полугруппа консервативных ортогональных подстановок удовлетворяет требованиям **Req1**–**Req3**. Согласно результатам статьи [6] отсюда следует

Утверждение 8. *Задача минимизации конечных детерминированных автоматов-преобразователей над полугруппой консервативных ортогональных подстановок может быть решена за время, полиномиальное относительно размера минимизируемого автомата.*

Опираясь на утверждение 8 и принимая во внимание тот способ трансляции стандартных схем программ в автоматы-преобразователи над полугруппами подстановок, который описан в утверждении 5, мы можем описать один из классов схем программ, для которого задача минимизации решается за полиномиальное время.

Теорема 4. *Задача минимизации решается за полиномиальное время в классе стандартных схем программ, удовлетворяющих двум условиям:*

1) *все дуги схемы помечены консервативными ортогональными подстановками;*

2) *каждому узлу схемы приписан атом, содержащий вхождения всех переменных из множества X.*

6. Заключение

Нам удалось показать, что задачи минимизации стандартных схем программ и автоматов-преобразователей, работающих над полугруппой подстановок, тесно взаимосвязаны между собой — методы решения одной из этих задач могут быть адаптированы для решения другой. В частности, эффективный метод минимизации автоматов-преобразователей, предложенный в статье [6], можно использовать при посредничестве стандартных схем программ для решения задачи оптимизации последовательных императивных программ. В то же время разрешимость проблемы логико-термальной эквивалентности стандартных схем программ гарантирует разрешимость задач проверки эквивалентности и минимизации для детерминированных автоматов-преобразователей, работающих над полугруппами подстановок.

Вопрос о сложности задач минимизации стандартных схем программ и автоматов-преобразователей остается открытым и требует дальнейшего исследования. Метод минимизации автоматов-преобразователей, предложенный в статье [6], применим только в том случае, когда полугруппа состояний данных, над которой работают преобразователи, удовлетворяет требованиям **Req1**–**Req3**. Как было показано в упомянутой статье, для полугрупп такого вида каждый класс эквивалентности автоматов-преобразователей содержит единственный минимиальный преобразователь. Обычно в теории автоматов неоднозначность решения задачи минимизации

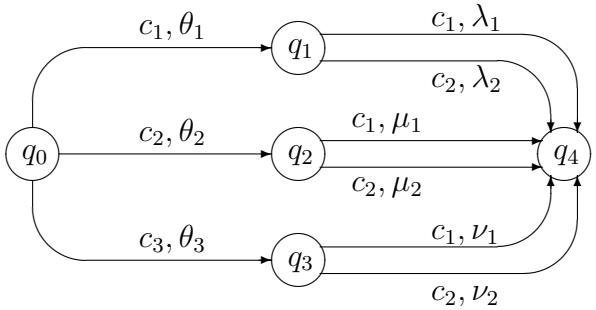

Рис. 1. Автомат-преобразователь П

Fig. 1. Transducer П

служит признаком того, что эта задача имеет большую вычислительную сложность. Так, например, однозначность решения задачи минимизации детерминированных конечных автоматов сопутствует быстрым алгоритмам минимизации автоматов этого вида, тогда как неоднозначность решения той же задачи для недетерминированных конечных автоматов сопровождается PSPACE-полнотой этой задачи в указанном классе автоматов. В заключение мы приведем простой пример, показывающий, что задача минимизации автоматов-преобразователей произвольного вида, работающих над полугруппой подстановок, может не иметь однозначного решения.

Рассмотрим автомат-преобразователь П, система переходов которого изображена на рис. 1. Его начальным состоянием является состояние q_0, а состоянием выхода — q_4. Действиями преобразователя служат подстановки над множеством переменных $X = \{x, y\}$ и функциональных символов $\mathcal{F} = \{f, g, h_1, h_2\}$. Действиями инициализации и финализации преобразователя служит тождественная подстановка $\varepsilon = \{x/x, y/y\}$. Прочие действия заданы следующими подстановками:

$$\theta_1 = \{x/g(f(x), g(x, x)), y = f(f(g(x, x)))\},$$
$$\theta_2 = \{x/f(f(x)), y/f(x)\},$$
$$\theta_3 = \{x/g(x, y), y/f(y)\},$$
$$\lambda_1 = \{x/h_1(x), y/h_1(y)\}, \qquad \lambda_2 = \{x/h_2(x), y/h_2(y)\},$$
$$\mu_1 = \{x/h_1(g(x, y)), y/h_1(f(f(y)))\}, \qquad \mu_2 = \{x/h_2(g(x, y)), y/h_2(f(f(y)))\},$$
$$\nu_1 = \{x/h_1(g(f(f(x))), f(x)), y = h_1(f(f(x)))\},$$
$$\nu_2 = \{x/h_2(g(f(f(x))), f(x)), y = h_2(f(f(x)))\}.$$

Непосредственной проверкой можно убедиться в том, что автомат-преобразователь П эквивалентен каждому из двух автоматов-преобразователей П′ и П″, которые заданы системами переходов, изображенными на рис. 2. В этих преобразователях используются действия, которые заданы подстановками

$$\xi_1' = \{x/x, y/g(x, x)\},\ \xi_2' = \{x/f(x), y/f(x)\},$$
$$\tau_i' = \{x/h_i(g(f(x), y)), y/h_i(f(f(y)))\}, \qquad i = 1, 2,$$
$$\xi_1'' = \{x/x, y/f(x)\},\ \xi_2'' = \{x/g(x, y), y/g(x, y)\},$$
$$\tau_i'' = \{x/h_i(g(f(f(x)), f(x))), y/h_i(f(f(y)))\}, \qquad i = 1, 2.$$

Автоматы-преобразователи П′ и П″ неизоморфны. Они являются минимальными преобразователями, поскольку никакой автомат-преобразователь с тремя состояни-

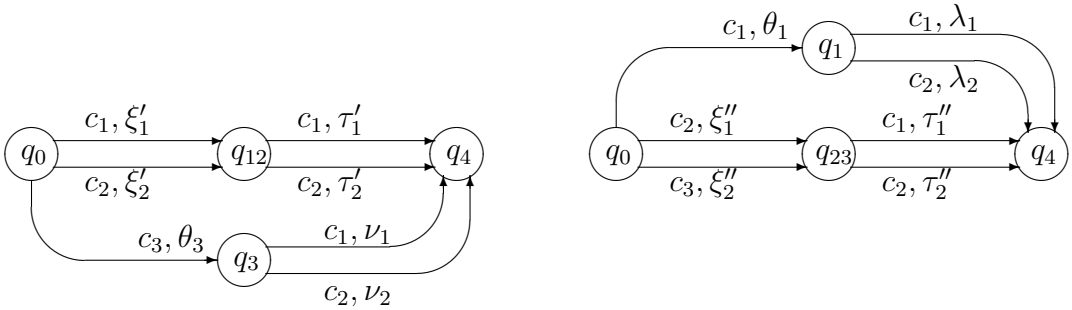

Рис. 2. Автоматы-преобразователи Π' и Π''

Fig. 2. Transducers Π' and Π''

ями им не эквивалентен. Таким образом, задача минимизации конечных детерминированных автоматов-преобразователей над полугруппой подстановок в общем случае может не иметь единственного решения. Поиск оценки вычислительной сложности этой задачи и разработка эффективных процедур ее решения составляют тему дальнейшего исследования.

И напоследок обратим внимание на то, что мы ограничились рассмотрением самой простой меры сложности стандартных схем программ — количеством узлов в графе, представляющем схему. Можно также оценивать сложность схемы суммарной сложностью всех термов в подстановках, приписанных дугам схемы. Задача минимизации схем относительно этой меры сложности также заслуживает внимания, и она, вероятно, может оказаться более трудной для решения.

Список литературы / References

[1] Глушков В. М., Летичевский А. А., “Теория дискретных преобразователей”, *Избранные вопросы алгебры и логики*, Новосибирск, 1973, 5–39; [Glushkov V. M., Letichevskij A. A., “Teoriya diskretnyh preobrazovatelej”, *Izbrannye voprosy algebry i logiki*, Novosibirsk, 1973, 5–39, (in Russian).]

[2] Ершов А. П., “Сведение задачи экономии памяти при составлении программ к задаче раскраски вершин графа”, *Доклады АН СССР*, **142**:4 (1962), 785–787; [Ershov A. P., “Svedenie zadachi ekonomii pamyati pri sostavlenii programm k zadache raskraski vershin grafa”, *Doklady AN SSSR*, **142**:4 (1962), 785–787, (in Russian).]

[3] Ершов А. П., “Современное состояние теории схем программ”, *Проблемы кибернетики*, **27**, 1973, 87–110; [Ershov A. P., “Sovremennoe sostoyanie teorii skhem program”, *Problemy kibernetiki*, **27** (1973), 87–110, (in Russian).]

[4] Захаров В. А., Новикова Т. А., “Полиномиальный по времени алгоритм проверки логико-термальной эквивалентности программ ”, *Труды Института системного программирования РАН*, **22** (2012), 435–455; [Zakharov V. A., Novikova T. A., “Polynomial time algorithm for checking strong equivalence of program”, *Proceedings of the Institute for System Programming*, **22** (2012), 435–455, (in Russian).]

[5] Захаров В. А., Подымов В. В., “Применение алгоритмов проверки эквивалентности для оптимизации программ”, *Труды Института системного программирования РАН*, **27**:4 (2015), 145–174; [Zakharov V. A., Podymov V. V., “On the application of equivalence checking algorithms for program minimization”, *Proceedings of the Institute for System Programming*, **27**:4 (2015), 145–174, (in Russian).]

[6] Захаров В. А., Темербекова Г. Г., “О минимизации конечных автоматов-преобразователей над полугруппами ”, *Моделирование и анализ информационных систем*, **23**:6 (2016), 741–753; [Zakharov V. A., Temerbekova G. G., “On the

Minimization of Finite State Transducers over Semigroups", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 741–753, (in Russian).]

[7] Захаров В. А., *Проблема эквивалентности программ: модели, алгоритмы, сложность*, М., 2016, 304 с.; [Zakharov V. A., *Problema ekvivalentnosti programm: modeli, algoritmy, slozhnost*, Moskva, 2016, 304 pp., (in Russian).]

[8] Иткин В. Э., "Логико-термальная эквивалентность схем программ", *Кибернетика*, 1972, № 1, 5–27; [Itkin V. E., "Logiko-termalnaya ekvivalentnost skhem program", *Cybernetics*, 1972, № 1, 5–27, (in Russian).]

[9] Котов В. Е., Сабельфельд В. К., *Теория схем программ*, Наука, М., 1991, 248 с.; [Kotov V. E., Sabel'fel'd V. K., *Teoriya skhem program*, Nauka, Moskva, 1991, 248 pp., (in Russian).]

[10] Криницкий Н. А., *Равносильные преобразования алгоритмов и программирование*, М., 1970, 304 с.; [Krinitskiy N. A., *Ravnosilnye preobrazovaniya algoritmov i programmirovanie*, Moskva, 1970, 304 pp., (in Russian).]

[11] Лавров С. С., "Об экономии памяти в замкнутых операторных схемах", *Журнал вычислительной математики и математической физики*, **1**:4 (1961), 687–701; English transl.: Lavrov S. S., "Store economy in closed operator schemes", *U.S.S.R. Comput. Math. Math. Phys.*, **1**:3 (1962), 810–828.

[12] Ляпунов А. А., "О логических схемах программ", *Проблемы кибернетики*, **1**, 1958, 46–74; [Lyapunov A. A., "O logicheskikh skhemakh program", *Problemy kibernetiki*, **1**, 1958, 46–74, (in Russian).]

[13] Подловченко Р. И., "Модели последовательных программ, применяемые для изучения функциональной эквивалентности программ", *Кибернетика*, 1979, № 1, 20–28; English transl.: Podlovchenko R. I., "Models of sequential programs used to study functional equivalence of programs", *Cybern Syst Anal*, **15**:1 (1979), 22–31.

[14] Янов Ю. И., "О логических схемах алгоритмов", *Проблемы кибернетики*, **1**, 1958, 75–127; [Yanov Yu. I., "O logicheskikh skhemakh algoritmov", *Problemy kibernetiki*, **1**, 1958, 75–127, (in Russian).]

[15] Eder E., "Properties of substitutions and unifications", *Journal of Symbolic Computations*, **1**:1 (1985), 31–46.

[16] Ershov A. P., "Alpha — an automatic programming system of high efficiency", *Journal of the Association for Computing Machinary*, **13**:1 (1966), 17–24.

[17] Friedman E. P., "Equivalence problems for deterministic languages and monadic recursion schemes", *Journal of Computer System Science*, **14**:3 (1977), 362–399.

[18] Harju T., Karhumaki J., "The equivalence of multi-tape finite automata", *Theoretical Computer Science*, **78**:2 (1991), 347–355.

[19] Luckham D. C., Park D. M., Paterson M. S., "On formalized computer programs", *Journal of Computer and System Science*, **4**:3 (1970), 220–249.

[20] Mohri M., "Minimization algorithms for sequential transducers", *Theoretical Computer Science*, **234** (2000), 177–201.

[21] Muchnick S., *Advanced Compiler Design and Implementation*, 1997, 1–856.

[22] Rutledge J. D., "Program schemata as automata", *Proc. of the 11-th Annual Symposium on Switching and Automata Theory (SWAT 1970)*, 1970, 7–24.

[23] Sabelfeld V. K., "The logic-termal equivalence is polynomial-time decidable", *Information Processing Letters*, **10**:2 (1980), 57–62.

[24] Senizergues G., "The equivalence problem for deterministic pushdown automata is decidable", *Proc. of the 24-th International Colloquium on Automata, Languages, and Programming (ICALP-1997). Lecture Notes in Computer Science*, **1256** (1997), 671–681.

[25] Shofrutt C., "Minimizing subsequential transducers: a survey", *Theoretical Computer Science*, **292** (2003), 131–143.

[26] Watson B. W., "A taxonomy of finite automata minimization algorithm", *Computing Science Report. Eindhoven University of Technology*, **93/44** (2005), 1–32.

[27] Zakharov V. A., "On the decidability of the equivalence problem for orthogonal sequential programs", *Grammars*, **2**:3 (1999), 271–281.

[28] Zakharov V. A., "Equivalence checking problem for finite state transducers over semigroups", *Proc. of the 6-th International Conference on Algebraic Informatics (CAI-2015). Lecture Notes in Computer Science*, **9270** (2015), 208–221.

Zakharov V. A., Zhailauova S. R., "On the Minimization Problem for Sequential Programs", *Modeling and Analysis of Information Systems*, **24**:4 (2017), 415–433.

DOI: 10.18255/1818-1015-2017-4-415-433

Abstract. First-order program schemata is one of the simplest models of sequential imperative programs intended for solving verification and optimization problems. We consider the decidable relation of logical-thermal equivalence of these schemata and the problem of their size minimization while preserving logical-thermal equivalence. We prove that this problem is decidable. Further we show that the first-order program schemata supplied with logical-thermal equivalence and finite state deterministic transducers operating over substitutions are mutually translated into each other. This relationship implies that the equivalence checking problem and the minimization problem for these transducers are also decidable. In addition, on the basis of the discovered relationship, we have found a subclass of first-order program schemata such that their minimization can be performed in polynomial time by means of known techniques for minimization of finite state transducers operating over semigroups. Finally, we demonstrate that in general case the minimization problem for finite state transducers over semigroups may have several non-isomorphic solutions.

Keywords: sequential program, transducer, minimization, substitution, semigroup, equivalence

On the authors:
Vladimir A. Zakharov, orcid.org/0000-0002-3794-9565, PhD, senior researcher
National Reserach University Higher School of Economics, Faculty of Computer Science
20 Myasnitskaya ul., Moscow 101000, Russia, e-mail: zakh@cs.msu.ru

Shynar R. Zhailauova, graduate student,
Lomonosov Moscow State University, Faculty of Computational Mathematics and Cybernetics,
Leninskiye Gory, GSP-1, 1-52, Moscow 119991, Russia, e-mail: gulgaisha93@mail.ru

Acknowledgments:
This work is supported by the by RFBR grants №16-01-00546 and №15-01-05742.